ASSIGNMENT 8:

Name: Om Prasanna Kulkarni

PRN: 22510034

Batch: S3

Branch: CSE

Topic: Python list and Dictionary

Q1) Write a Python script to perform below operations on list

a. Initialize a list lst=[ 1.5, “Python”, -5, 4, 0.8, -3.2, ‘C++’]

b. Add 0.8 to the list

c. Add [4,1.5,6,0.8] to the existing list

d. Insert any element at position 2.

e. Remove ‘C++’ from the list.

f. Remove element at position 5.

g. Find out the index of element 5.

h. Find out the occurrence of element 4.

i. Slice the list from 2 nd to 6 th position.

j. Reverse the list.

k. Create a copy of this list and display.

Source Code:  
lst = [1.5, "Python", -5, 4, 0.8, -3.2, 'C++']

lst.append(0.8)

lst.extend([4, 1.5, 6, 0.8])

lst.insert(2, "NewElement")

if 'C++' in lst:

    lst.remove('C++')

if len(lst) > 5:

    del lst[5]

index\_of\_5 = lst.index(5) if 5 in lst else -1

occurrence\_of\_4 = lst.count(4)

sliced\_lst = lst[1:6]

lst.reverse()

lst\_copy = lst.copy()

print("Modified List: ", lst)

print("Index of 5: ", index\_of\_5)

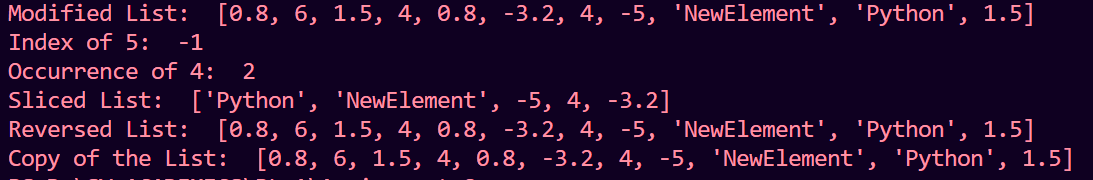
print("Occurrence of 4: ", occurrence\_of\_4)

print("Sliced List: ", sliced\_lst)

print("Reversed List: ", lst)

print("Copy of the List: ", lst\_copy)

OUTPUT:



Q2) Create a function in python to perform swapping of two numbers using

1. Temporary variable

Source Code:

def swap\_with\_temp(a, b):

    temp = a

    a = b

    b = temp

    return a, b

num1 = 5

num2 = 10

num1, num2 = swap\_with\_temp(num1, num2)

print("After swapping (with temp): num1 =", num1, "num2 =", num2)

Output:



b.No Temporary Variable

Source Code:

def swap\_without\_temp(a, b):

    a = a + b

    b = a - b

    a = a - b

    return a, b

num1 = 5

num2 = 10

num1, num2 = swap\_without\_temp(num1, num2)

print("After swapping (without temp): num1 =", num1, "num2 =", num2)

OUTPUT:



Q3) Create a function in python to accept and return multiple arguments.

Source Code:

def add\_numbers(a, b):

    return a + b

result = add\_numbers(3, 4)

print(result)

OUTPUT:



Q4) Inventory Management: In a retail store, use a list to keep track of product stock

levels, and implement functions to add, remove, and update items in the

inventory.

Source Code:

inventory = []

product1 = {'name': 'Widget', 'stock': 50}

product2 = {'name': 'Gadget', 'stock': 30}

inventory.append(product1)

inventory.append(product2)

inventory.remove(product1)

product2['stock'] = 40

for product in inventory:

    print(f"Product: {product['name']}, Stock: {product['stock']}")

OUTPUT:



Q5) To-Do List Application: Create a to-do list application where users can add,

remove, and update tasks using a list.

Source Code:  
  
todo\_list = []

def display\_todo\_list():

    if not todo\_list:

        print("Your to-do list is empty.")

    else:

        print("To-Do List:")

        for i, task in enumerate(todo\_list, 1):

            print(f"{i}. {task}")

def add\_task(task):

    todo\_list.append(task)

    print(f"Task '{task}' added to your to-do list.")

def remove\_task(task\_number):

    if 1 <= task\_number <= len(todo\_list):

        removed\_task = todo\_list.pop(task\_number - 1)

        print(f"Task '{removed\_task}' removed from your to-do list.")

    else:

        print("Invalid task number.")

def update\_task(task\_number, new\_task):

    if 1 <= task\_number <= len(todo\_list):

        todo\_list[task\_number - 1] = new\_task

        print(f"Task {task\_number} updated to '{new\_task}'.")

    else:

        print("Invalid task number.")

while True:

    print("\nOptions:")

    print("1. Display To-Do List")

    print("2. Add Task")

    print("3. Remove Task")

    print("4. Update Task")

    print("5. Exit")

    choice = input("Enter your choice: ")

    if choice == '1':

        display\_todo\_list()

    elif choice == '2':

        task = input("Enter the task: ")

        add\_task(task)

    elif choice == '3':

        task\_number = int(input("Enter the task number to remove: "))

        remove\_task(task\_number)

    elif choice == '4':

        task\_number = int(input("Enter the task number to update: "))

        new\_task = input("Enter the updated task: ")

        update\_task(task\_number, new\_task)

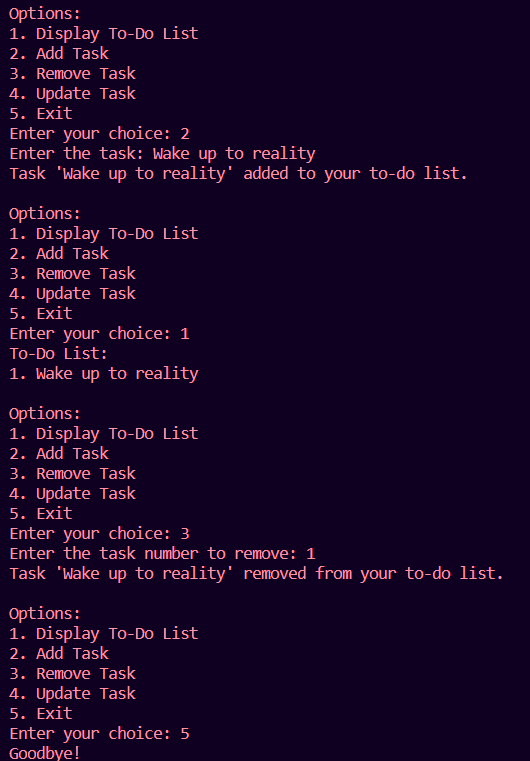
    elif choice == '5':

        print("Goodbye!")

        break

    else:

        print("Invalid choice. Please select a valid option.")

OUTPUT:  


Q6) Social Media Metrics: Analyze engagement metrics from social media posts.

Use a list to store the number of likes, comments, and shares for each post and

calculate the average engagement.

Source Code:

post\_metrics = [

    [100, 20, 10],

    [75, 15, 5],

    [120, 30, 12],

    [90, 18, 9],

    [80, 22, 8]

]

def calculate\_average\_engagement(metrics\_list):

    total\_likes = 0

    total\_comments = 0

    total\_shares = 0

    for post in metrics\_list:

        total\_likes += post[0]

        total\_comments += post[1]

        total\_shares += post[2]

    total\_posts = len(metrics\_list)

    average\_likes = total\_likes / total\_posts

    average\_comments = total\_comments / total\_posts

    average\_shares = total\_shares / total\_posts

    return average\_likes, average\_comments, average\_shares

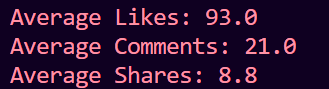
avg\_likes, avg\_comments, avg\_shares = calculate\_average\_engagement(post\_metrics)

print(f"Average Likes: {avg\_likes}")

print(f"Average Comments: {avg\_comments}")

print(f"Average Shares: {avg\_shares}")

OUTPUT:



Q7) Student Gradebook: Design a program to manage student grades. Store student

names and their corresponding test scores in a list and perform operations like

calculating the average score.

Source Code:

student\_grades = [

    {"name": "Alice", "score": 85},

    {"name": "Bob", "score": 92},

    {"name": "Charlie", "score": 78},

    {"name": "David", "score": 89},

    {"name": "Eve", "score": 95}

]

total\_score = sum(student['score'] for student in student\_grades)

average\_score = total\_score / len(student\_grades)

print(f"Average Score: {average\_score}")

OUTPUT:  


Q8) Event Registration: Develop a system to manage event registrations. Use a list

to store attendee names, and implement functionalities like checking in

attendees and generating an attendee list.

Source Code:

attendee\_list = []

def register\_attendee(name):

    attendee\_list.append(name)

    print(f"{name} has been registered for the event.")

def check\_in\_attendee(name):

    if name in attendee\_list:

        print(f"{name} has been checked in.")

    else:

        print(f"{name} is not registered for the event.")

def generate\_attendee\_list():

    print("Attendee List:")

    for i, attendee in enumerate(attendee\_list, 1):

        print(f"{i}. {attendee}")

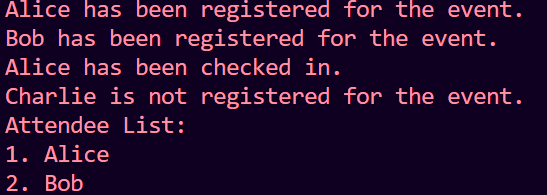
register\_attendee("Alice")

register\_attendee("Bob")

check\_in\_attendee("Alice")

check\_in\_attendee("Charlie")

generate\_attendee\_list()

OUTPUT:  
  


Q9) Write a Python script to perform below operations on Dictionary

a. Create a dictionary

b. Print keys and values of dictionary

c. Search for a specific key, if found, display its value, if not display default

value.

d. Add new key and value

e. Remove any element

f. Copy this dictionary

g. Find out length of dictionary

h. Delete the dictionary

Source Code:

my\_dictionary = {

    "name": "Alice",

    "age": 25,

    "country": "USA"

}

for key, value in my\_dictionary.items():

    print(f"Key: {key}, Value: {value}")

key\_to\_search = "age"

default\_value = "Key not found"

print(f"Value of key {key\_to\_search}: {my\_dictionary.get(key\_to\_search, default\_value)}")

my\_dictionary["occupation"] = "Software Engineer"

key\_to\_remove = "country"

if key\_to\_remove in my\_dictionary:

    del my\_dictionary[key\_to\_remove]

copied\_dictionary = my\_dictionary.copy()

length\_of\_dictionary = len(my\_dictionary)

print(f"Length of the dictionary: {length\_of\_dictionary}")

my\_dictionary.clear()

OUTPUT:

